8-3

УДК 510.1

***В.К. Задирака, А.Н. Терещенко***
Институт кибернетики имени В.М. Глушкова НАНУ, г. Киев, Украина
Zvk140@ukr.net

# Использование арифметики многорозрядных чисел в современных компьютерных технологиях решения задач трансвычислительной сложности

В работе приводится ряд оригинальных эффективных по быстродействию алгоритмов выполнения операций над многоразрядными числами и их применение для решения задач двухключевой криптографии и для высокоточных вычислений с помощью компьютерной технологии решения задач прикладной и вычислительной математики с заданными значениями характеристик качества по точности и быстродействию.

## Введение

В настоящее время существует много прикладных задач: в двухключевой криптографии (шифрование, расшифрование, электронная цифровая подпись, криптографические протоколы), моделировании физических, химических (биохимических) процессов, аэродинамики, гидродинамики, расчета оболочек ядреных реакторов, при решении которых активно используется арифметика многоразрядных чисел [1], [2].

Если раньше арифметика многоразрядных чисел в основном использовалась в двухключевой криптографии для обеспечения необходимой криптостойкости и производительности соответствующих алгоритмов (для записи одного числа требуется от 1024 до 4096 бит), то в последние годы, в связи с появлением суперкомпьютеров и решением на них сверхсложных задач, область применения многоразрядной арифметики значительно расширилась. В последнем случае это связано с обязательным исследованием наследственной погрешности и погрешности округления, накапливаемых в ходе длительных численных расчетов. Неучёт этих погрешностей приводит к тому, что иногда получают компьютерные решения, не обладающие физическим смыслом.

Достоверность получаемых результатов в значительной мере зависит от чувствительности задачи к малым ошибкам округлений. Например, для системы линейных алгебраических уравнений понятия «хорошо и плохо обусловленная матрица» тесно связаны с вычислительными возможностями конкретного компьютера и длиной машинного слова. В результате одна и та же система может квалифицироваться для одной длины мантиссы машинного слова как «машинно плохо обусловленная» или «почти вырожденная», а для другой – «машинно хорошо обусловленная» [3]. Таким образом, один из резервов для получения достоверного решения для плохо обусловленных задач – повышение точности представления чисел и выполнения операций.

**Цель статьи** – с одной стороны, привлечь внимание к существенному расширению области применения многоразрядной арифметики, изложению новых эффективных алгоритмов выполнения операций над многоразрядными числами, а с другой – показать место этой арифметики (как одного из резервов оптимизации вычислений) в современ-

ных компьютерных технологиях решения задач дискретной, прикладной и вычислительной математики с заданными значениями характеристики качества по точности и быстродействию [1].

# 1 Некоторые новые эффективные по быстродействию алгоритмы выполнения операций над многоразрядными числами

## 1.1 Построение алгоритмов вычисления многоразрядной свертки

Постановка задачи. Если целые положительные числа $U_N$, $V_N$, $C_N$ вида

$$U_N = \sum_{i=0}^{N-1} u_i 2^{\omega i}, \; V_N = \sum_{i=0}^{N-1} v_i 2^{\omega i}, \; C_N = \sum_{i=0}^{N-1} c_i 2^{\omega i},$$

где $0 \le u_i, v_i, c_i < 2^{\omega}$, $N = 2^n$, тогда выражение $C_N$ вида

$$C_N = U_N \otimes V_N = \left( \sum_{i=0}^{N-1} u_i 2^{\omega i} \right) \otimes \left( \sum_{i=0}^{N-1} v_i 2^{\omega i} \right) = \sum_{\tau=0}^{N-1} c_\tau 2^{\omega \tau}, \qquad (1)$$

где $c_\tau = \sum_{\rho+\delta=\tau (\operatorname{mod} N)} u_\rho \cdot v_\delta$ представляет собой циклическую свертку $U_N$ и $V_N$.

**Вычисление свертки с использованием быстрого преобразования Уолша (БПУ).** Проиллюстрируем метод использования преобразования Уолша для вычисления сверток на примере вычисления свертки длины $N = 4$. При этом достаточно выполнить 5 операций умножения

$$X_4 = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \; X_4 = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \; W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}; \; T_4 = \begin{bmatrix} 0 \\ -t \\ 0 \\ +t \end{bmatrix},$$

$$t = (x_1 - x_3) \cdot (y_0 - y_2) = x_1 y_0 + x_3 y_2 - x_1 y_2 - x_3 y_0, \; \tilde{X}_4 = (W_4 \cdot X_4) \cdot (W_4 \cdot Y_4);$$

$$R_4 = \frac{1}{4} W_4 \cdot \tilde{X}_4 + T_4. \qquad (2)$$

Результат (2) представляет собой циклическую свертку последовательностей $X_4$ и $Y_4$: $R_4 = X_4 \otimes Y_4$. Видно, что свертку длиной $N = 4$ нельзя вычислить с использованием только БПУ. Необходимо использовать корректирующий элемент $t$. Далее показано, что для вычисления более длинных сверток необходимо большее число корректирующих элементов. Последовательность всех корректирующих элементов можно представить в виде корректирующего вектора $T$ и существует простой способ его построения.

**Операторы вычисления**. Введем следующие операторы *E*, *O*, *L*, *H*, *U*, *D*, *W*, *A*, *S*, (*E-Even* (четный), *O-Odd* (нечетный), *L-Low* (младшие), *H-High* (старшие), *U-Up* (вверх), *D-Down* (вниз), *W-Walsh* (Уолш), *A-Add* (добавление), *S-Subtract* (вычитание) и определим их следующим образом:

$$(EX_N)_k = x_{2k}, \; (OX_N)_k = x_{2k+1}, (LX_N)_k = x_k, \; (HX_N)_k = x_{N/2+k}, \; k = \overline{0, N/2-1};$$

циклический сдвиг элементов вверх $V_N = UX_N$, $v_k = x_{\langle k+1 \rangle_N}$, $k = \overline{0, N-1}$;

циклический сдвиг элементов вниз $V_N = DX_N$, $v_k = x_{\langle k+N-1 \rangle_N}$, $k = \overline{0, N-1}$;

циклический сдвиг элементов вверх $p$ раз $V_N = U^p\left(X_N\right)$;

циклический сдвиг элементов вниз $p$ раз $V_N = D^p\left(X_N\right)$.

Проиллюстрируем вид операторов для случая $N = 8$:

$$EX_8 = \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix},\ OX_8 = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix},\ LX_8 = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix},\ HX_8 = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix},\ UY_4 = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_0 \end{bmatrix},\ DY_4 = \begin{bmatrix} y_3 \\ y_0 \\ y_1 \\ y_2 \end{bmatrix}.$$

Последовательное выполнение операторов $O$, $W$, $H$ обозначим оператором $S$, выполнение операторов $O$, $W$, $L$ – через $A$:

$$SX_N = H(W(OX_N)) = E(OX_N) - O(OX_N),$$

$$AY_N = L(W(OY_N)) = E(OY_N) + O(OY_N).$$

Вычисление свертки разрядностью $N = 8$ с использованием БПУ и с учетом введенных выше операторов можно выразить следующим образом:

$$\overset{€}{X}_8 = W_8 \cdot X_8,\ \overset{€}{Y}_8 = W_8 \cdot Y_8,\ \overset{€}{X}_4 = L\overset{€}{X}_8 - H\overset{€}{X}_8,\ Y_4 = U\left(EY_8\right) - EY_8,\ \overset{€}{Y}_4 = W_4 \cdot Y_4;$$

$$\overset{€}{A}_8 = \overset{€}{X}_8 \cdot \overset{€}{Y}_8,\ \overset{€}{A}_4 = \overset{€}{X}_4 \cdot \overset{€}{Y}_4,\ \overset{€}{T}_2 = S\overset{€}{X}_8 \cdot A\overset{€}{Y}_8,\ \overset{€}{T}_1 = S\overset{€}{X}_4 \cdot A\overset{€}{Y}_4;$$

$$A_8 = W_8 \cdot \overset{€}{A}_8,\ A_4 = W_4 \cdot \overset{€}{A}_4,\ \overline{t}_2 = W_2 \cdot \overset{€}{T}_2;$$

$$OA_8 = OA_8 + A_4,\ OT_2 = OT_2 + T_1,$$

$$\begin{bmatrix} LLR_8 \\ \cdots \\ HLR_8 \\ \cdots \\ LHR_8 \\ \cdots \\ HHR_8 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} LLA_8 \\ \cdots \\ HLA_8 \\ \cdots \\ LHA_8 \\ \cdots \\ HHA_8 \end{bmatrix} + \frac{1}{8}\begin{bmatrix} 0 \\ \cdots \\ -T_2 \\ \cdots \\ 0 \\ \cdots \\ +T_2 \end{bmatrix}.$$

Из приведенных формул видно, что для вычисления свертки разрядностью $N = 8$ необходимо вычислить три БПУ разрядностью $N = 8$, два БПУ разрядностью $N = 4$, одно БПУ разрядностью $N = 2$, а также 15 операций однословного умножения.

**Теорема 1**. Для вычисления свертки $R_4 = X_4 \otimes Y_4$ длиной $N = 2^n$ с использованием алгоритма БПУ необходимо перемножить поэлементно два вектора длиной $5 \cdot 3^{n-2}$ каждый.

**Лемма 1**. Количество операций сложения и вычитания для вычисления БПУ в векторе $X$, разбитого на основные и дополнительные секции с длинами $K$, оценивается следующим соотношением $Q_N^{\pm} = (10n + 4)3^{n-3}$, где $N = 2^n$.

**Пошаговый алгоритм умножения двух чисел разрядностью $N = 2^n$ через свертку, используя БПУ.** Введем дополнительные обозначения.

$X = \left(X, Z\right)$ обозначает, что к элементам $X$ добавляются элементы $Z$ справа. Соответственно длина $X$ увеличивается на число элементов вектора $Z$.

$x_i$ обозначает $i$-й элемент $X$. При этом $X$ считается полнозаполненным без учета разбивки на секции разрядностью степени двойки.

$\left(X\right)_i$ обозначает, что в $X$ рассматривается $i$-я секция. Элементами $K$ являются длины секций $X$. Соответственно элемент $k_i$ содержит длину $i$-й секции в $X$.

$\left[\left(X\right)_i\right]_j$ обозначает, что в $i$-й секции $X$ берется $j$-й элемент. Вектор $K$ содержит длины секций в $X$.

$\left(X_M\right)_i$ обозначает, что в векторе $X$ рассматривается $i$-я секция. Считается, что $X$ разбит на секции равной длины $M$, где $M$ – степень двойки, $\phi$ – пустой вектор.

Последовательность операций разбивается символом «;», которые выполняются в обычном порядке слева направо. Операции внутри символа «;», разделенные запятой, выполняются в обратном порядке справа налево. Если операцию обозначить номером ее выполнения, то получим строку, представляющую последовательность выполнения операций: 2, 1; 6, 5, 4, 3; 8, 7.

С учетом введенных обозначений опишем следующий алгоритм.

**Алгоритм 1.** Реализация операции умножения двух чисел разрядностью $N = 2^n$ через свертку, использующую БПУ.

**Шаг 1.** Предвычислить:

$$S = \phi,\ K = \left(2^n\right);\ S = \left(S, 2^j, S\right),\ K = \left(K, \left(K/2\right)\right),\ j = \overline{0, n-3}.$$

Добавить длины дополнительных секций: $K = \left(K, \left(K/4\right)\right)$, $S = \left(0, S, 0, S\right)$.

Инициализировать: $X = \left(X_{N=2^n}\right)_0 = X_N$; $Y = \left(Y_{N=2^n}\right)_0 = Y_N$.

**Шаг 2.** БПУ начальной секции $\left(X\right)_0$: $\tilde{X}_N = W_N \cdot X_N$.

**Шаг 3.** Представить ДПУ остальных секций $X$ как линейных комбинаций секций $X$:

$$X = \left(X, \left(L\left(X_M\right)_i - H\left(X_M\right)_i\right)\right),\ i = \overline{0, 3^j - 1},\ M = 2^{n-j},\ j = \overline{0, n-3}.$$

**Шаг 4.** Вычислить БПУ секций $Y$ за исключением двух последних шагов.

$$\left(Y_M\right)_i = WV,\ Y = \left(Y, \left(U\left(EV\right) - EV\right)\right), V = \left(Y_M\right)_i,\ i = \overline{0, 3^j - 1},\ M = 2^{n-j},\ j = \overline{0, n-3}.$$

**Шаг 5.** Вычислить последние два шага БПУ. $\left(Y_4\right)_i = W\left(Y_4\right)_i$, $i = \overline{0, 3^{n-2} - 1}$;

$$\left(Y_2\right)_i = W\left(Y_2\right)_i,\ i = \overline{0, (2 \cdot 3^{n-2}) - 1}.$$

**Шаг 6.** Сформировать дополнительных секций $X = \left(X, SX\right)$, $Y = \left(Y, AY\right)$.

**Шаг 7.** Вычислить. $R = X \cdot Y$.

**Шаг 8.** Вычислить ОБПУ. $\left(R\right)_i = W_{k_i} \cdot \left(R\right)_i, i = \overline{0, N/2 - 1}$.

**Шаг 9.** Вычислить начальные основные и дополнительные секции:

$$\left[\left(R\right)_{j-s}\right]_{s\cdot(2m+1)+i} += \left[\left(R\right)_j\right]_{s\cdot m+i}, i = \overline{0, s-1}, m = \overline{0, (k_j/s) - 1}, s = s_j,$$

$$j = \overline{(N/2 - 1)..(N/4 + 1), (N/4 - 1)..1}.$$

**Шаг 10.** Корректировать начальную основную секцию:

$$\left[\left(R\right)_0\right]_{N/4+i} -= \left[\left(R\right)_{N/4}\right]_i, i = \overline{0, N/4 - 1};$$

$$\left[\left(R\right)_0\right]_{3N/4+i} += \left[\left(R\right)_{N/4}\right]_i, i = \overline{0, N/4 - 1}.$$

**Лемма 2.** Оценка общего числа операций сложения и вычитания в алгоритме 1 определяется соотношением $Q_N^{\pm} \le n2^n - 11 \cdot 2^{n-2} + (18n + 43)3^{n-3}$.

**Теорема 2.** Оценка количества вычислительных затрат, необходимых для вычисления свертки (1) с помощью алгоритма 1 на основе БПУ, может быть оценена соотношением $Q \le Q^{\pm}_{FWT} + Q^{*}_{FWT}$, где $Q^{\pm}_{FWT} \le n2^{n} - 11\cdot 2^{n-2} + (18n+43)3^{n-3}$ – оценка общего количества необходимых операций сложения и вычитания, $Q^{*}_{FWT} \le 5\cdot 3^{n-2}$ – оценка количества необходимых умножений ($FWT$ – Fast Walsh transform).

Ниже приведена табл. 1, содержащая число умножений, необходимых для вычисления циклической свертки различными алгоритмами (классическим, с использованием БПФ и алгоритмом 1 на основе БПУ). Известно, что для вычисления свертки стандартным методом необходимо выполнить порядка $N^2$ операций умножения и сложения. Для косвенного алгоритма вычисления свертки $R_k$ с использованием алгоритма БПФ асимптотические оценки количества необходимых умножений и сложений с плавающей точкой в случае $N = 2^n, n > 0$ – целое, имеют соответственно вид [4], [5]: $Q^{*}_{FFT} \le 9n\cdot 2^{n-1}$, $Q^{+}_{FFT} \le 13{,}5n\cdot 2^{n-1}$ ($FFT$ – Fast Fourier transform).

Оценки $Q^{+}_{FWT}$ и $Q^{*}_{FWT}$ определены теоремой 2.

Таблица 1

| $n$ | $Q^{*}_{ST}$ | $Q^{*}_{FFT}$ | $Q^{*}_{FWT}$ | $Q^{+}_{FFT}$ | $Q^{+}_{FWT}$ |
|---|---|---|---|---|---|
| 5 | 1,024 | 720 | 135 | 1,080 | 1,287 |
| 6 | 4,096 | 1,890 | 405 | 2,592 | 4,339 |
| 7 | 16,384 | 4,032 | 1,215 | 6,048 | 14,395 |
| 8 | 65,536 | 9,216 | 3,645 | 13,824 | 47,271 |
| 9 | 262,144 | 20,736 | 10,935 | 31,104 | 154,103 |
| 10 | 1,048,576 | 46,080 | 32,805 | 69,120 | 499,499 |
| 11 | 4,194,304 | 101,376 | 98,415 | 152,064 | 1,611,219 |
| 12 | 16,777,216 | 221,184 | 295,245 | 331,776 | 5,175,151 |

Отметим, что $Q^{*}_{FFT}$, $Q^{+}_{FFT}$ содержит число операций с плавающей точкой, $Q^{*}_{FWT}$, $Q^{+}_{FWT}$ – число операций над целыми числами. На современных компьютерах операции с плавающей точкой выполняются в 10 и более раз медленнее, чем операции над целыми числами.

Отметим, что, как показал сравнительный анализ сложности, рассматриваемый алгоритм 1 при $n < 11$ требует меньше вычислительных затрат, чем алгоритм, использующий БПФ. Поэтому его целесообразно применять для вычисления сверток разрядности $N \le 2^{10}$.

Коэффициенты ускорений $\delta^{*}$ и $\delta^{\pm}$ по отношению к операциям действительного умножения и сложения для исследуемого алгоритма по сравнению с алгоритмом секционирования на основе БПФ [4] имеет вид

$$\delta^{*} = k_f \cdot \frac{9n\cdot 2^{n-1}}{5\cdot 3^{n-2}},\ \delta^{\pm} = k_f \cdot \frac{13{,}5n\cdot 2^{n-1}}{2^{n}\left(n - 11/4\right) + (18n+43)3^{n-3}},$$

где в $k_f = t_f / t_i$, $t_f, t_i$ – времена выполнения операции с плавающей точкой и время выполнения операции над целыми числами соответственно. $k_f = 10$.

Ниже в табл. 2 приведены коэффициенты ускорения $\delta^{*}$ и $\delta^{\pm}$.

Таблица 2

| $n$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta^*$ | 72 | 72 | 64 | 53,3 | 42,7 | 33,2 | 25,3 | 18,9 | 14,0 | 10,3 | 7,49 |
| $\delta^\pm$ | 23,1 | 16,4 | 11,8 | 8,51 | 6,05 | 4,25 | 2,95 | 2,04 | 1,39 | 0,95 | 0,65 |

Анализ таблицы и предыдущие рассуждения показывают, что алгоритм 1 целесообразен для вычисления свертки длиной $N = 2^n, n < 11$.

**Усовершенствованный метод Питасси – Девиса.** Предлагается метод вычисления свертки разрядностью $N = K \cdot 2^n$, где $K$ – нечетное, который **расширяет диапазон разрядностей**, в котором существуют эффективные методы вычисления циклической свертки. Далее показано, что для сверток такой разрядности достаточно вычислить две свертки разрядностью $K$, в отличие от метода Питасси – Девиса, в котором на каждой итерации необходимо вычислять три свертки. При этом используется свойство нечетной разрядности: результат непарного циклического сдвига в одну сторону равняется парному сдвигу в другую сторону.

**Лемма 3.** Для циклической свертки любой разрядности имеют место соотношения:

$$UX_N \otimes UY_N = X_N \otimes Y_N,$$

$$X_N \otimes UY_N = U(X_N \otimes Y_N),$$

$$UX_N \otimes Y_N = X_N \otimes DY_N.$$

В общем виде предлагаются формулы вычисления свертки разрядностью $N = K \cdot 2^n$, где $K$ – нечетное, $p = \lfloor K/2 \rfloor$:

$$A_{N/2} = \left(EX_N + U^p(OX_N)\right) \otimes \left(EY_N + U^p(OY_N)\right), \quad ER_N = 1/2\left(A_{N/2} + S_{N/2}\right),$$
$$S_{N/2} = \left(EX_N - U^p(OX_N)\right) \otimes \left(EY_N - U^p(OY_N)\right), \quad OR_N = 1/2\left(A_{N/2} - S_{N/2}\right),$$

$$\begin{array}{c|c:c} EY_N & EX_N & U^p(OX_N) \\ \hdashline U^p(OY_N) & U^p(OX_N) & EX_N \\ \hline & ER_N & U^p(OR_N) \end{array}.$$

С учетом того, что необходимо 10 умножений для вычисления свертки разрядностью $N = 5$, далее показано, что достаточно 20 умножений для вычисления свертки разрядностью $N = 10 = 2 \cdot 5$. В этом случае предыдущие формулы принимают вид:

$$A_5 = \left(EX_{10} + U^2(OX_{10})\right) \otimes \left(EY_{10} + U^2(OY_{10})\right), \quad ER_{10} = 1/2\left(A_5 + S_5\right),$$
$$S_5 = \left(EX_{10} - U^2(OX_{10})\right) \otimes \left(EY_{10} - U^2(OY_{10})\right), \quad OR_{10} = 1/2\left(A_5 - S_5\right),$$

$$\begin{array}{c|c:c} EY_{10} & EX_{10} & U^2(OX_{10}) \\ \hdashline U^2(OY_{10}) & U^2(OX_{10}) & EX_{10} \\ \hline & ER_{10} & U^2(OR_{10}) \end{array}.$$

Из формул видно, что необходимо циклически сдвинуть нечетные элементы на две позиции и использовать всего две матрицы меньшей разрядности

$$EX_{10} = \begin{bmatrix} x_0 & x_8 & x_6 & x_4 & x_2 \\ x_2 & x_0 & x_8 & x_6 & x_4 \\ x_4 & x_2 & x_0 & x_8 & x_6 \\ x_6 & x_4 & x_2 & x_0 & x_8 \\ x_8 & x_6 & x_4 & x_2 & x_0 \end{bmatrix}, \quad U^2(OX_{10}) = \begin{bmatrix} x_5 & x_3 & x_1 & x_9 & x_7 \\ x_7 & x_5 & x_3 & x_1 & x_9 \\ x_9 & x_7 & x_5 & x_3 & x_1 \\ x_1 & x_9 & x_7 & x_5 & x_3 \\ x_3 & x_1 & x_9 & x_7 & x_5 \end{bmatrix}$$

для представления свертки разрядностью $N = 10 = 2 \cdot 5$:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $EY_{10}$ | $y_0$ | $x_0$ | $x_8$ | $x_6$ | $x_4$ | $x_2$ | $x_5$ | $x_3$ | $x_1$ | $x_9$ | $x_7$ |
| | $y_2$ | $x_2$ | $x_0$ | $x_8$ | $x_6$ | $x_4$ | $x_7$ | $x_5$ | $x_3$ | $x_1$ | $x_9$ |
| | $y_4$ | $x_4$ | $x_2$ | $x_0$ | $x_8$ | $x_6$ | $x_9$ | $x_7$ | $x_5$ | $x_3$ | $x_1$ |
| | $y_6$ | $x_6$ | $x_4$ | $x_2$ | $x_0$ | $x_8$ | $x_1$ | $x_9$ | $x_7$ | $x_5$ | $x_3$ |
| | $y_8$ | $x_8$ | $x_6$ | $x_4$ | $x_2$ | $x_0$ | $x_3$ | $x_1$ | $x_9$ | $x_7$ | $x_5$ |
| $U^2(OY_{10})$ | $y_5$ | $x_5$ | $x_3$ | $x_1$ | $x_9$ | $x_7$ | $x_0$ | $x_8$ | $x_6$ | $x_4$ | $x_2$ |
| | $y_7$ | $x_7$ | $x_5$ | $x_3$ | $x_1$ | $x_9$ | $x_2$ | $x_0$ | $x_8$ | $x_6$ | $x_4$ |
| | $y_9$ | $x_9$ | $x_7$ | $x_5$ | $x_3$ | $x_1$ | $x_4$ | $x_2$ | $x_0$ | $x_8$ | $x_6$ |
| | $y_1$ | $x_1$ | $x_9$ | $x_7$ | $x_5$ | $x_3$ | $x_6$ | $x_4$ | $x_2$ | $x_0$ | $x_8$ |
| | $y_3$ | $x_3$ | $x_1$ | $x_9$ | $x_7$ | $x_5$ | $x_8$ | $x_6$ | $x_4$ | $x_2$ | $x_0$ |
| | | $r_0$ | $r_2$ | $r_4$ | $r_6$ | $r_8$ | $r_5$ | $r_7$ | $r_9$ | $r_1$ | $r_3$ |
| | | $ER_{10}$ | | | | | $U^2(OR_{10})$ | | | | |

Рисунок 1 – Вычисление свертки длиной $N = 10$ предложенным методом

Таблица 3 – Количество операций умножения при вычислении свертки разрядностью $N = K \cdot 2^n$, $n > 1$, где $K = 3, 5, 7, 9$

| Значения $K$ | Количество операций умножения $O^*(K)$ для вычисления свертки разрядностью $K$ | Разрядность свертки $N = K \cdot 2^n$, $n > 1$ | Количество операций умножения $O^*(N)$ при вычислении свертки разрядностью $N = K \cdot 2^n$, $n > 1$ |
|---|---|---|---|
| 3 | 4 | $N = 3 \cdot 2^n$ | $O^*(N) \le 8 \cdot 3^{n-1}$ |
| 5 | 10 | $N = 5 \cdot 2^n$ | $O^*(N) \le 20 \cdot 3^{n-1}$ |
| 7 | 16 | $N = 7 \cdot 2^n$ | $O^*(N) \le 32 \cdot 3^{n-1}$ |
| 9 | 19 | $N = 9 \cdot 2^n$ | $O^*(N) \le 38 \cdot 3^{n-1}$ |

## 1.2 Построение алгоритмов умножения двух многоразрядных чисел

Постановка задачи. Если целые положительные числа $U_N$ и $V_N$ вида $U_N = \sum_{i=0}^{N-1} u_i 2^{\omega i}$, $V_N = \sum_{i=0}^{N-1} v_i 2^{\omega i}$, где $0 \le u_i, v_i < 2^{\omega}$, $N = 2^n$, тогда выражение $R_{2N}$ вида

$$R_{2N} = U_N \cdot V_N = \left( \sum_{i=0}^{N-1} u_i 2^{\omega i} \right) \cdot \left( \sum_{i=0}^{N-1} v_i 2^{\omega i} \right) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} u_i v_j 2^{\omega(i+j)} = \sum_{k=0}^{2N-1} r_k 2^{\omega k}$$

представляет результат умножения двух многоразрядных чисел $U_N$ и $V_N$.

Необходимо построить эффективные алгоритмы вычисления $R_{2N}$.

**Реализация операции многоразрядного умножения с использованием циклической свертки.** В качестве примера рассмотрим умножение двух чисел

$U_4 = (u_0, u_1, u_2, u_3)$, $V_4 = (v_0, v_1, v_2, v_3)$ разрядностью 4 с использованием свертки. Алгоритм можно представить в виде:

$$R_8 = (u_0, u_1, u_2, u_3, 0,0,0,0) \otimes (0,0,0,0, u_3, u_2, u_1, u_0).$$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $u_0$ | 0 | 0 | 0 | 0 | $v_3$ | $v_2$ | $v_1$ | $v_0$ |
| $u_1$ | 0 | 0 | 0 | $v_3$ | $v_2$ | $v_1$ | $v_0$ | 0 |
| $u_2$ | 0 | 0 | $v_3$ | $v_2$ | $v_1$ | $v_0$ | 0 | 0 |
| $u_3$ | 0 | $v_3$ | $v_2$ | $v_1$ | $v_0$ | 0 | 0 | 0 |
| 0 | $v_3$ | $v_2$ | $v_1$ | $v_0$ | 0 | 0 | 0 | 0 |
| 0 | $v_2$ | $v_1$ | $v_0$ | 0 | 0 | 0 | 0 | $v_3$ |
| 0 | $v_1$ | $v_0$ | 0 | 0 | 0 | 0 | $v_3$ | $v_2$ |
| 0 | $v_0$ | 0 | 0 | 0 | 0 | $v_3$ | $v_2$ | $v_1$ |
| | $r_0$ | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ |

Рисунок 2 – Умножение в столбик двух чисел разрядностью 4 с использованием свертки

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $u_0$ | 0 | 0 | 0 | 0 | $v_3$ | $v_2$ | $v_1$ | $v_0$ |
| $u_1$ | 0 | 0 | 0 | $v_3$ | $v_2$ | $v_1$ | $v_0$ | 0 |
| $u_2$ | 0 | 0 | $v_3$ | $v_2$ | $v_1$ | $v_0$ | 0 | 0 |
| $u_3$ | 0 | $v_3$ | $v_2$ | $v_1$ | $v_0$ | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | $r_0$ | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ |

Рисунок 3 – Умножение в столбик двух чисел разрядностью 4 с использованием свертки с учетом нулевых строк

На рис. 3, в отличие от рис. 2, учтено то, что умножение нулевых элементов из левого столбика на элементы матрицы дает в результате ноль.

Окончательно получаем:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $u_0$ | | | | | $v_3$ | $v_2$ | $v_1$ | $v_0$ |
| $u_1$ | | | | $v_3$ | $v_2$ | $v_1$ | $v_0$ | |
| $u_2$ | | | $v_3$ | $v_2$ | $v_1$ | $v_0$ | | |
| $u_3$ | | $v_3$ | $v_2$ | $v_1$ | $v_0$ | | | |
| | $r_0$ | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ | $r_7$ |

Рисунок 4 – Умножение в столбик двух чисел разрядностью 4 с использованием свертки без нулевых строк

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | $u_3$ | $u_2$ | $u_1$ | $u_0$ |
| | | | | $v_3$ | $v_2$ | $v_1$ | $u_0$ |
| | | | | $u_0v_3$ | $u_0v_2$ | $u_0v_1$ | $u_0v_0$ |
| | | | $u_1v_3$ | $u_1v_2$ | $u_1v_1$ | $u_1v_0$ | |
| | | $u_2v_3$ | $u_2v_2$ | $u_2v_1$ | $u_2v_0$ | | |
| | $u_3v_3$ | $u_3v_2$ | $u_3v_1$ | $u_3v_0$ | | | |
| $r_7$ | $r_6$ | $r_5$ | $r_4$ | $r_3$ | $r_2$ | $r_1$ | $r_0$ |

Рисунок 5 – Умножение двух чисел в столбик

Из рис. 4 и 5 видно, что использованная таким образом свертка реализует стандартный метод умножения в столбик, за исключением того, что результат получается в обратном порядке.

**Алгоритм 1**. Реализация операции умножения двух $N$-разрядных чисел $U_N = \sum_{i=0}^{N-1} u_i 2^{\omega i}$ и $V_N = \sum_{i=0}^{N-1} v_i 2^{\omega i}$, где $\omega = 8,16,32$, с использованием свертки длиной $2N$:

**Шаг 1**. $u_i = 0$, $i = \overline{N, 2N-1}$.

**Шаг 2**. $v_{2N-1-i} = v_i$, $i = \overline{0, N-1}$; $v_i = 0$, $i = \overline{N, 2N-1}$.

**Шаг 3**. $R_{2N} = U_{2N} \otimes V_{2N}$.

**Шаг 4**. $R_{2N} = R_{2N}(2N-1-i)$, $i = \overline{0, 2N-1}$ или $(r_i, r_{2N-1-i}) = (r_{2N-1-i}, r_i)$, $i = \overline{0, N-1}$.

**Шаг 5**. $r_i = Lr_i$, $r_{i+1} = r_{i+1} + Hr_i$, $i = \overline{0,2N-2}$.

Вектор $R_{2N}$ разрядностью $2N$ будет содержать результат операции умножения чисел $U_N$ и $V_N$ разрядностью $N$. Операции $Lr_i$ и $Hr_i$ обозначают, что берется младшая и старшая часть $r_i$ соответственно.

**Модифицированный диагональный метод**, который использует рекуррентные соотношения и идею Карацубы-Офмана для умножения многоразрядных чисел, выражается следующим соотношением:

$$U_N \cdot V_N = \sum_{i=0}^{N-1} u_i 2^{\omega i} \cdot \sum_{i=0}^{N-1} v_i 2^{\omega i} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} u_i v_i 2^{\omega(i+j)} + \sum_{i=1}^{N-1}\sum_{j=0}^{i-1} \left(u_i - u_j\right)\cdot\left(v_j - v_i\right) 2^{\omega(i+j)}. \quad (3)$$

С учетом резервов оптимизации для первого слагаемого в (3) и максимального использования суммирования по диагонали на регистрах эта формула примет следующий вид:

$$U_N \cdot V_N = W_0 + \sum_{k=1}^{2N-3}\left(W_k + \sum_{i=0}^{\lceil k/2\rceil-1}\left(u_{k-i} - u_i\right)\left(v_i - v_{k-i}\right)\right)2^{\varpi k} + W_{2N-2} 2^{\varpi(2N-2)},$$

$$W_0 = u_0 v_0,$$

$$W_i = W_{i-1} + u_i v_i,\ i = \overline{1, N-1},$$

$$W_i = W_{i-1} - u_{i-N} v_{i-N},\ i = \overline{N, 2N-3},$$

$$W_{2N-2} = u_{N-1} v_{N-1}. \quad (4)$$

Формулы (3), (4) при $N = 2$ совпадают с формулой Карацубы-Офмана [6]:

$$U_2 \cdot V_2 = \sum_{i=0}^{1} u_i 2^{\omega i} \cdot \sum_{i=0}^{1} v_i 2^{\omega i} = u_0 v_0 + u_0 v_0 2^{\omega} + u_1 v_1 2^{\omega} + u_1 v_1 2^{\omega 2} + \left(u_1 - u_0\right)\cdot\left(v_0 - v_1\right)2^{\omega}.$$

$$U_2 \cdot V_2 = W_0 + \left(W_1 + \left(u_1 - u_0\right)\left(v_0 - v_1\right)\right)2^{\varpi} + W_2 2^{\varpi 2},\ W_0 = u_0 v_0,\ W_1 = W_0 + u_1 v_1,\ W_2 = u_1 v_1.$$

Априорные оценки сложности выполнения операции умножения $N$-разрядных чисел составляют

$$C^*(N) \le \frac{1}{2}N^2 + \frac{3}{2}N - 1,\ C^+(N) \le \frac{1}{2}N^2 + \frac{1}{2}N - 1,\ C^-(N) \le N^2 - 2,$$

где $C^*(N)$ – количество однословных операций умножения, $C^+(N)$ – количество двухсловных операций сложения, $C^-(N)$ – количество однословных операций вычитания.

Если все $N$ разрядов в $U_N$ разбиваются на одинаковые числа, которые кодируются блоками, то сумма $\sum_{i=0}^{\lceil k/2\rceil-1}\left(u_{k-i} - u_i\right)\left(v_i - v_{k-i}\right)$ в (4) всегда будет равна нулю и количество операций умножения будет линейно зависеть от длины множителя $C^*(N) = 2N - 2$ и формула (4) примет вид:

$$U_N \cdot V_N = \sum_{k=0}^{2N-2} W_k 2^{\varpi k},$$

$$W_0 = uv_0;\ W_k = W_{k-1} + uv_k,\ i = \overline{1, N-1};\ W_k = W_{k-1} - uv_{k-N},\ k = \overline{N, 2N-3};\ W_{2N-2} = uv_{N-1}.$$

Очевидно, что если и $V_N$ состоит из одинаковых по значению слов, то достаточно только одной операции однословного умножения: $C^*(N) = 1$.

Если представить числа $U_N$ и $V_N$ в виде последовательности битов и существует возможность разбиения на одинаковые по значению битовые последовательности мень-

шей длины, то в этом случае не обязательно разбивать $U_N$ и $V_N$ таким образом, чтобы длины таких меньших последовательностей совпадали по длине для каждого из разрядов в $U_N$ и $V_N$ соответственно.

Формулу (3) удобно использовать для случая, когда количество разрядов одного множителя кратно количеству разрядов другого множителя $U_N \cdot V_{kN}$, что значительно уменьшает сложность выполнения многоразрядного умножения для такого случая.

**Умножение больших *N*-разрядных чисел с вычислением только *N*-разрядных ДПФ.** При умножении двух $N$-разрядных чисел получается $2N$-разрядный результат, поэтому необходимо вычислять $2N$-разрядную свертку и соответственно $2N$-разрядные ДПФ множителей, которые рассматриваются как входные $2N$-разрядные дискретные действительные сигналы. Далее приведен метод, при котором достаточно вычислять только $N$-разрядные ДПФ. Для этого приводятся формулы перехода от разрядности $N$ к разрядности $2N$ и наоборот. Данный метод позволяет уменьшить число комплексных операций умножения приблизительно в два раза по сравнению со стандартным алгоритмом умножения с использованием БПФ, при котором необходимо вычислять все $2N$ значений свертки.

Вначале приведем некоторые свойства ДПФ действительного сигнала, которые далее будут использоваться в данной статье [7].

**Лемма 4.** Если $X_{2N}(r)$ и $\hat{X}_{2N}(r)$, $r=\overline{0,\,2N-1}$ – действительный сигнал и его ДПФ, то $\hat{X}_{2N}(N+r)=\hat{X}^*_{2N}(N-r)$, $r=\overline{1,\,N-1}$, и $\hat{X}_{2N}(N)$ – действительное число.

**Лемма 5.** Если $X_{2N}(r), Y_{2N}(r)$ и $\hat{X}_{2N}(r), \hat{Y}_{2N}(r)$, $r=\overline{0,\,2N-1}$ – действительные сигналы и их ДПФ соответственно, то сигнал $\hat{I}_{2N}(r)=\hat{X}_{2N}(r)\cdot\hat{Y}_{2N}(r)$, $r=\overline{0,\,2N-1}$, обладает следующим свойством $\hat{I}_{2N}(r)=\hat{I}^*_{2N}(2N-r)$, $r=\overline{1,\,N-1}$.

**Лемма 6 (формулы распаковки).** Если $U_{2N}(r)$, $r=\overline{0,\,2N-1}$ – действительный сигнал, то ДПФ $\hat{U}_{2N}(r)$, $\hat{U}_N(r)$ сигналов $U_{2N}(r)$, $r=\overline{0,\,2N-1}$, и $U_N(r)=P_N(r)+iQ_N(r)$ ($P_N(r)=U_{2N}(2r)$, $Q_N(r)=U_{2N}(2r+1)$), $r=\overline{0,\,N-1}$, связаны следующими соотношениями:

$$\hat{U}_{2N}(N-r)=\hat{P}^*_N(r)-\left(W^r_{2N}\hat{Q}_N(r)\right)^*,\ r=\overline{1,\,N/2-1},\ \hat{U}_{2N}(0)=\mathrm{Re}\,\hat{U}_N(0)+\mathrm{Im}\,\hat{U}_N(0),$$

$$\hat{U}_{2N}(N)=\mathrm{Re}\,\hat{U}_N(0)-\mathrm{Im}\,\hat{U}_N(0),\ \hat{U}_{2N}(N/2)=\hat{U}^*_N(N/2).$$

**Лемма 7 (формулы упаковки).** Если $R_{2N}(r)$, $r=\overline{0,\,2N-1}$ – действительный сигнал, то ДПФ $\hat{R}_N(r)$, $\hat{R}_{2N}(r)$ сигналов $R_N(r)=P_N(r)+iQ_N(r)$ ($P_N(r)=R_{2N}(2r)$, $Q_N(r)=R_{2N}(2r)$), $r=\overline{0,\,N-1}$, связаны следующими соотношениями

$$\hat{P}_N(r)=\frac{1}{2}\left[\hat{R}_{2N}(r)+\hat{R}^*_{2N}(N-r)\right],\ \hat{Q}_N(r)=\frac{W^{-r}_{2N}}{2}\left[\hat{R}_{2N}(r)-\hat{R}^*_{2N}(N-r)\right]\ r=\overline{1,\,N/2-1},$$

$$\hat{R}_N(N-r)=\hat{P}^*_N(r)+i\hat{Q}^*_N(r),\ r=\overline{1,\,N/2-1};$$

$$\hat{R}_N(0)=\frac{\hat{R}_{2N}(0)+\hat{R}_{2N}(N)}{2}+i\,\frac{\hat{R}_{2N}(0)-\hat{R}_{2N}(N)}{2},\ \hat{R}_N(N/2)=\hat{R}^*_{2N}(N/2).$$

**Алгоритм 3**. Приведем пошаговое описание алгоритма умножения двух *N*-разрядных чисел с вычислением только *N*-разрядных ДПФ.

**Шаг 1.** Добавить $N$ старших нулей к сигналам $U_N$, $V_N$ $U_N(r+N)=V_N(r+N)=0$, $r=\overline{0,N-1}$ и из полученных сигналов $U_{2N}$, $V_{2N}$ сформировать сигналы $X_N(r)=U_N(2r)+$ $+iU_N(2r+1)$, $Y_N(r)=V_N(2r)+iV_N(2r+1)$, $r=\overline{0,\ N-1}$.

**Шаг 2.** Вычислить ДПФ $\hat{X}_N(r)$, $\hat{Y}_N(r)$, $r=\overline{0,\ N-1}$.

**Шаг 3.** Перейти от разрядности $N$ к разрядности $2N$-распаковка. Вычислить $N+1$ первые элементы $\hat{X}_{2N}(r),\hat{Y}_{2N}(r)$, $r=\overline{0,\ N}$ (Лемма 6).

**Шаг 4.** Вычислить $N+1$ первые значения $2N$-разрядной свертки $\hat{I}_{2N}(r)=$ $=\hat{X}_{2N}(r)\cdot\hat{Y}_{2N}(r)$, $r=\overline{0,\ N}$.

**Шаг 5.** Перейти от разрядности $2N$ к разрядности $N$-упаковка. Вычисление $\hat{I}_N(r)$ из $\hat{I}_{2N}(r)$ (Лемма 7).

**Шаг 6.** Вычислить ОДПФ $I_N(r), r=\overline{0,\ N-1}$.

**Шаг 7.** Найти искомый сигнал $U_{2N}(r)$, $r=\overline{0,\ 2N-1}$ (результат умножения чисел $U_N(r),V_N(r)$, $r=\overline{0,N-1}$), используя выражения $R_{2N}(2r)=\mathrm{Re}I_N(r)/N$, $R_{2N}(2r+1)=\mathrm{Im}I_N(r)/N$, $r=\overline{0,\ N-1}$.

Тестирование проводилось согласно методике, описанной в [8] на процессоре Pentium IV 2.4 GHz.

С учетом того, что число комплексных операций умножения для вычисления $N$- и $2N$-разрядного ДПФ соответственно $k_N=N\log_2 N$ и $k_{2N}=2N\log_2 2N$, найдем априорный коэффициент ускорения:

$$k=\frac{k_{2N}}{k_N}=\frac{2N\log_2 2N}{N\log_2 N}=\frac{2(\log_2 2+\log_2 N)}{\log_2 N}=2+\frac{2}{\log_2 N}.$$

Таблица 4 – Результаты тестирования приведенного алгоритма по сложности по сравнению со стандартным алгоритмом на основе БПФ

| Кол-во байтов | Время вычисления, с | | Коэффициент ускорения |
|---|---|---|---|
| | $2N$ | $N$ | |
| 384 | 14,06 | 5,687 | 2,472 |
| 768 | 7,17 | 3,70 | 1,938 |
| 1536 | 3.77 | 1,78 | 2,118 |
| 3072 | 2.054 | 0,97 | 2,118 |

Для примера вычислим $k$ при 3072 байтов. С учетом того, что предложенный алгоритм оперирует 3-байтовыми значениями, получаем

$$k_{3072}=2+\frac{2}{\log_2 1024}=2{,}2.$$

В алгоритме использовались следующие резервы оптимизации вычислений:

1. Таблица коэффициентов преобразования Фурье $N/4$ может быть уменьшена в два раза за счет того, что синусы матрицы вычисляются из косинусов. Достаточно хранить только косинусы в матрице, что уменьшает время работы с памятью.

2. Мантисса сопроцессора используется только на треть при вычислении ДПФ базовым методом на 32-битных процессорах, то есть один разряд ДПФ отвечает одному байту. Предлагается рассматривать 3 байта как один байт, что максимально использует

сопроцессор. Использование 64-битных процессоров дает возможность представлять один разряд свертки в 7 байтах и вычислять операции умножения 1024-битных чисел, используя всего лишь 32-разрядную циклическую свертку ( $7 \cdot 8 \cdot 32 = 1792 > 1024$ ).

3. Перегруппировка обрабатываемых данных улучшает быстродействие метода за счет увеличения операций последовательного доступа (ускорение на 11%). При этом элементы действительной и комплексной частей располагаются вместе, а не в отдельных массивах.

4. Использование динамического распределения памяти увеличивает разрядность множителя до 6144 байт (ускорение на 20%).

Таблица 5 – Коэффициенты ускорения алгоритма Шенхаге-Штрассена по времени (с) с базовым алгоритмом умножения до проведенной оптимизации

| *N/k* | 166МГц | 400Мгц |
|---|---|---|
| | | |
| 512 | 16 | 8.2 |
| 1024 | 10 | 4.4 |
| 2048 | 5.5 | 2.3 |
| 4096 | 3.0 | 1.4 |
| 8192 | 1.7 | 0.75 |
| 16384 | 0.99 | 0.42 |
| 32768 | 0.56 | 0.23 |

Таблица 6 – Коэффициенты ускорения алгоритма Шенхаге-Штрассена по времени (с) с базовым алгоритмом умножения после проведенной оптимизации

| *N/k* | 166МГц | 400Мгц |
|---|---|---|
| 768 | 4.5 | 4.4 |
| 1536 | 2.4 | 2.5 |
| 3072 | 1.6 | 1.4 |
| 6144 | 1.01 | 0.8 |
| 12288 | 0.7 | 0.4 |
| 24576 | 0.4 | 0.23 |
| 49115 | 0.23 | 0.13 |
| 98230 | 0.14 | 0.07 |

**Алгоритм умножения без умножения.** Основное отличие предлагаемого метода состоит в том, что при его реализации операция умножения не используется вообще, а вычислительный процесс основывается только на сложении. Метод напоминает собой классический метод умножения в столбик. Для вычисления результата умножения используется таблица предвычислений из девяти элементов. Каждый элемент таблицы предвычислений равен первому множителю, умноженному на порядковый номер в таблице предвычислений. При нахождении результата происходит последовательная выборка разрядов второго множителя. Значения $i$-го разряда соответствуют номеру, по которому выбирается из таблицы предвычислений элемент, который прибавляется к результату, начиная с $i$-го разряда. Нулевые разряды пропускаются.

Для малого числа $N$, когда значения разрядов множителя повторяются редко, использование данного метода нецелесообразно. Этот метод применим при больших $N$. В этом случае вероятность появления цифры от 0 до 9 в каждом разряде приблизительно равна 10%.

При практической реализации на ЭВМ необходимо учитывать, что большие числа рассматриваются как последовательность $N$ байт, которые являются основной ячейкой памяти и могут содержать любые значения от 0 до 255. Так как для компьютера удобнее использовать 16-ричную систему исчисления, то байт лучше представить в виде двух 16-ричных цифр (старшего и младшего). Тогда таблица $X_{15}$ будет иметь 15 элементов и процедуру получения произведения $R_{2N}$ можно разбить на два этапа. На 1-м этапе для умножения используются только младшие разряды всех байт числа $V_N$. На втором этапе таблица $X_{15}$ переформатируется умножением каждого элемента на 16, что представляет собой сдвиг на 4 бита. Затем таблица $X_{15}$ используется для умножения старших разрядов всех байт числа $V_N$.

**Алгоритм 5.** Приведем пошаговое описание алгоритма умножения чисел $U_N$ и $V_N$, каждое из которых содержит $N$ байт (ω = 8).

**Шаг 1.**Вычислить таблицу $X_{15}$: $X_1 = \sum_{j=0}^{N-1} u_j 2^{\omega j}$; $X_i = X_{i-1} + X_1, i = \overline{2,15}$.

**Шаг 2.** Вычислить результат умножения числа $U_N$ на младшие разряды числа $V_N$: $R_{2N} = \sum_{i=0}^{N-1} X_{(Lv_i)} 2^{\omega i}$, где $Lv_i$ – младшие разряды $i$-го байта числа $V_N$.

**Шаг 3.** Корректировать таблицу $X_{15}$ для использования ее при получении результата умножения на старшие разряды числа $V_N$: $X_1 = X_1 + X_{15}$; $X_i = X_{i-1} + X_1$, $i = \overline{2,15}$.

**Шаг 4.** Найти результат умножения числа $U_N$ на младшие разряды числа $V_n$: $R_{2N} = \sum_{i=0}^{N-1} X_{(Hv_i)} 2^{\omega i}$, где $Hv_i$ – младшие разряды $i$-го байта числа $V_N$.

Накладные расходы для построения таблицы предвычислений при малых $N$ гораздо больше, чем остальные вычисления при обработке младших и старших разрядов.

Приведем описание некоторых подходов, которые были использованы при численной реализации предложенного метода.

1. Группировка операций чтения и записи в память.

2. Использование двух таблиц. Расчет двух таблиц предвычислений для старшего и младшего разрядов и одновременное добавление элементов этих таблиц к результату.

3. Использование кеша процессора. Последовательное добавление $i$-го элемента таблицы предвычислений к результату, начиная с номеров разрядов второго множителя, в которых встречается значение $i$, что позволяет максимально использовать кеш процессора.

На медленных компьютерах наилучшей будет модификация, при которой используются одновременно две таблицы предвычислений для старшего и младшего разрядов каждого байта числа. При увеличении мощности компьютеров наилучшей оказывается модификация, при которой используется последовательное добавление $i$-го элемента таблицы предвычислений.

Таблица 7 – Сравнительный анализ по быстродействию предложенных подходов по сравнению с диагональным методом на процессоре DX4-100

| $N$ | 1-я модификация (группировка) | 2-я модификация (две таблицы) | 3-я модификация (кеш) |
|---|---|---|---|
| 64 | 8.00 | 1,55 | 3.00 |
| 128 | 6,11 | 2,14 | 3,89 |
| 256 | 4,50 | 2.04 | 3.76 |
| 512 | 3,89 | 2,17 | 3,60 |
| 1024 | 4,09 | 3,54 | 3,58 |

Таблица 8 – Сравнительный анализ по быстродействию предложенных подходов по сравнению с диагональным методом на Celeron 1Ghz

| $N$ | 1-я модификация (группировка) | 2-я модификация (две таблицы) | 3-я модификация (кеш) |
|---|---|---|---|
| 64 | 8,80 | 11,00 | 7.80 |
| 128 | 5.07 | 6.29 | 4,93 |
| 256 | 4.97 | 6.11 | 4.87 |
| 512 | 4.68 | 6.03 | 4.62 |
| 1024 | 4.56 | 6.09 | 4.64 |

$C(N) \le 2N^2 + 30N + k_m(4N^2 + 91N)$ – оценка общего количества необходимых вычислительных затрат, где $k_m$ – коэффициент, который представляет соотношение времени выполнения операции чтения/записи в память и времени выполнения операции сложения для определенного класса компьютеров. Для процессоров типа Celeron, коэффициент $k_m \ge 2$. Тогда $C(N) \le 10N^2 + 212N$.

## 1.3 Операция возведения в многоразрядную степень по многоразрядному модулю

Постановка задачи. Если целые положительные числа $U_N$, $V_N$, $R_N$, $M_N$, $T_N$ вида $U_N = \sum_{i=0}^{N-1} u_i 2^{\omega i}$, $V_N = \sum_{i=0}^{N-1} v_i 2^{\omega i}$, $R_N = \sum_{i=0}^{N-1} r_i 2^{\omega i}$, $M_n = \sum_{i=0}^{n-1} m_i 2^{\omega i}$, $T_N = \sum_{i=0}^{N-1} t_i 2^{\omega i}$, где $0 \le u_i, v_i, r_i, m_i, t_i < 2^{\omega}$, тогда выражение $T_N$ вида

$$T_N = \left\langle \left(U_N \cdot V_N\right)^{R_N} \right\rangle_{M_N} = \left\langle \left( \left( \sum_{i=0}^{N-1} u_i 2^{\omega i} \right) \cdot \left( \sum_{i=0}^{N-1} v_i 2^{\omega i} \right) \right)^{\sum_{i=0}^{N-1} r_i 2^{\omega i}} \right\rangle_{M_N}$$

представляет результат возведения многоразрядных чисел $U_N$, $V_N$ в многоразрядную степень $R_N$ по многоразрядному модулю $M_N$.

**Новая реализация алгоритма умножения Монтгомери.** Алгоритм умножения Монтгомери используется для вычисления многоразрядного остатка и возведения в многоразрядную степень. Рассматривается новая реализация метода Монтгомери. В реализации введены и используются две системы сложных остатков, что позволяет использовать китайскую теорему об остатках. Вычисления происходят одновременно в двух системах остатков.

Показано, что в алгоритме Монтгомери задачу нахождения остатка умножения двух $N$-разрядных чисел по $N$-разрядному модулю можно привести к нахождению остатка результата однословных умножений по однословному модулю, что позволяет ускорить этот метод.

**Лемма 8.** Если модуль $M = \prod_{i=0}^{N-1} m_i$, такой, что $(m_j, m_i) = 1$, $i \ne j$ и известны числа $h_i$, такие, что $\langle h_i \cdot M_i \rangle_{m_i} = 1$, $M_i = \dfrac{M}{m_i}$, $(M_i, m_i) = 1$, $i = \overline{0, N-1}$, то для любого числа $U < M$ верны следующие соотношения

$$U = \left\langle \sum_{i=0}^{N-1} \left( \langle u_i h_i \rangle_{m_i} \cdot M_i \right) \right\rangle_M, \quad \sum_{i=0}^{N-1} \frac{\langle u_i h_i \rangle_{m_i}}{m_i} < N, \quad \sum_{i=0}^{N-1} \left( \langle u_i h_i \rangle_{m_i} \cdot M_i \right) < N \cdot M,$$

где $\langle U \rangle_{m_i} = u_i$, $i = \overline{0, N-1}$.

**Теорема 3** (переход из одной системы остаточных классов в другую). Пусть задано две системы остаточных классов $m$ и $m'$ с модулями $M = \prod_{i=0}^{N-1} m_i$ и $M' = \prod_{i=0}^{N-1} m'_i$, такие, что $(m_j, m_i) = 1$, $(m'_j, m'_i) = 1$, $i \ne j$ и число $U < M$ имеет следующие остатки: $\langle U \rangle_{m_i} = u_i$, $i = \overline{0, N-1}$. Тогда сложность представления числа $U$ в системе остаточных

классов с модулем $M'$, где $M < M'$, выражается следующими априорными оценками: $C^*(N) \le N^2 + 2N$, $C^{\div}(N) \le N$, $C^{\langle\ \rangle}(N) \le 2N$, где $C^*(N)$, $C^{\div}(N)$ – количества однословных операций умножения и деления, $C^{\langle\ \rangle}(N)$ – число операций вычисления однословного модуля.

Для алгоритма, приведенного ниже, необходимы предвычисления.

При помощи расширенного алгоритма Евклида находим числа $M^{-1}$ и $D^{-1}$, такие, что $M \cdot M^{-1} - D \cdot D^{-1} = 1$.

Представляем $D$ и $D^{-1}$ в остатках $d_i = \langle D \rangle_{m_i}$, $d_i' = \langle D \rangle_{m_i'}$, $d_i^{-1} = \langle D^{-1} \rangle_{m_i'}$.

Находим числа $h_i$, $h_i'$, такие, что $\langle h_i \cdot M_i \rangle_{m_i} = 1$, $M_i = M/m_i$, $(M_i, m_i) = 1$, $\langle h_i' \cdot M_i' \rangle_{m_i'} = 1$, $M_i' = M'/m_i'$, $(M_i', m_i') = 1$, $i = \overline{0, N-1}$.

Вычисляем матрицу $W'$ $\{w_{ij}' = \langle M_j \rangle_{m_i'}, i, j = \overline{0, N-1}\}$, используемую для перевода промежуточного результата из системы остаточных классов $m$ в систему классов $m'$. Также находим $m_i'' = \langle M \rangle_{m_i'}$, $i = \overline{0, N-1}$.

Аналогично вычисляется матрица $W$ $\{w_{ij} = \langle M_j' \rangle_{m_i}, i, j = \overline{0, N-1}\}$, используемая для обратного перевода промежуточного результата из системы классов $m'$ в систему классов $m$. Для этого также необходимо определить $m_i''' = \langle M' \rangle_{m_i}$, $i = \overline{0, N-1}$, представляющий модуль $M'$ в системе остатков $m_i$.

Операцию деления на число $M$ в системе остаточных классов $m'$ заменим умножением числа $M^{\div}$: $m_i'^{\div} = \langle M^{\div} \rangle_{m_i'}$, $\langle m_i'^{\div} \cdot m_i'' \rangle_{m_i'} = 1$, $m_i'' = \langle M \rangle_{m_i'}$, $i = \overline{0, N-1}$.

Найдем $N$-остатки чисел $U$ и $V$

$$\overline{U} = \langle U \cdot M \rangle_N, \ \overline{V} = \langle V \cdot M \rangle_N;$$

$$u_i = \langle \overline{U} \rangle_{m_i}, \ u_i' = \langle \overline{U} \rangle_{m_i'}, \ v_i = \langle \overline{V} \rangle_{m_i}, \ v_i' = \langle \overline{V} \rangle_{m_i'}, \ i = \overline{0, N-1}.$$

**Алгоритм 5.** Одновременное использование двух систем остаточных классов.
Приведем пошаговое описание алгоритма.

**Шаг 1.** Вычислить $t_i = \langle u_i v_i \rangle_{m_i}$, $t_i' = \langle u_i' v_i' \rangle_{m_i'}$, $i = \overline{0, N-1}$.

**Шаг 2.** Вычислить $s_i = \langle t_i d_i^{-1} \rangle_{m_i}$, $i = \overline{0, N-1}$.

**Шаг 3.** Перейти в систему остаточных классов $m'$.

**Шаг 3а**. Вычислить $f_i = \langle s_i h_i \rangle_{m_i}, i = \overline{0, N-1}$.

**Шаг 3б.** Вычислить $k = \left\lfloor \frac{1}{2^{\omega}} \sum_{i=0}^{N-1} \left\lfloor \frac{f_i \cdot 2^{\omega}}{m_i} \right\rfloor \right\rfloor$.

**Шаг 3в.** Вычислить $s_i' = \left\langle \sum_{j=0}^{N-1} w_{ij}' f_j \right\rangle_{m_i'}$, $s_i' = \langle s_i' - k m_i'' \rangle_{m_i'}, i = \overline{0, N-1}$.

**Шаг 4.** Вычислить $t_i' = \langle t_i' + d_i' s_i' \rangle_{m_i'}$, $i = \overline{0, N-1}$.

**Шаг 5.** Вычислить $t_i' = \langle t_i' m_i'^{\div} \rangle_{m_i'}$.

**Шаг 6.** Возвратиться в исходную систему остаточных классов $m$.

**Шаг 6а.** Вычислить $f_i' = \langle t_i' h_i' \rangle_{m_i'}, i = \overline{0, N-1}$.

**Шаг 6б.** Вычислить $k' = \left\lfloor \frac{1}{2^\omega} \sum_{i=0}^{N-1} \left\lfloor \frac{f_i' \cdot 2^\omega}{m_i'} \right\rfloor \right\rfloor$.

**Шаг 6в.** Вычислить $t_i = \left\langle \sum_{j=0}^{N-1} w_{ij} f_j' \right\rangle_{m_i}, t_i = \langle t_i - k' m_i'' \rangle_{m_i}, i = \overline{0, N-1}$.

**Шаг 7.** Результат образуется в ячейках $\langle t_i \rangle_{m_i}, \langle t_i' \rangle_{m_i'}$, $i = \overline{0, N-1}$.

**Теорема 4.** Если числа $\overline{U}_N$ и $\overline{V}_N$ $N$-разрядные, то количество операций в алгоритме 5 можно представить следующими априорными оценками: $C^*(N) \le 2N^2 + 9N$, $C^{\div}(N) \le 2N$, $C^{\langle\,\rangle}(N) \le 9N$, где $C^*(N)$, $C^{\div}(N)$ – количество однословных операций умножения и деления, $C^{\langle\,\rangle}(N)$ – количество операций вычисления остатка по однословному модулю.

# 2 Применение для решения прикладных задач

В связи с ограниченным объемом статьи остановимся тезисно на некоторых приложениях, приведенных в п. 1 результатов.

Первое, на что хотелось обратить внимание, это то, что предлагаемые выше алгоритмы являются новыми и, как показал сравнительный анализ, лучше известных алгоритмов на сегодняшний день по быстродействию.

Второе – для каждого из алгоритмов получены конструктивные оценки (априорные и апостериорные) сложности, что позволяет произвести качественный сравнительный анализ с ранее известными.

Третье – хотя большинство из приведенных алгоритмов, за исключением модификации алгоритма Монтгомери, осуществляют быстрое умножение многоразрядных чисел, учитывая, что возведение в степень состоит из операций умножения и возведения в квадрат, данные алгоритмы применимы и для решения этой более сложной задачи.

Теперь относительно других приложений приведенных результатов. Существует много классов задач вычислительной, прикладной и дискретной математики, для решения которых необходима техника вычислений, которая использует алгоритмы выполнения разных операций над многоразрядными числами. Среди таких задач можно указать задачи аппроксимации функций, моделирование физических, химических процессов, аэродинамики, гидродинамики, защиты информации и другие.

В связи с увеличением сложности решаемых задач, например, систем линейных алгебраических сравнений с числом неизвестных 33 – 35 миллионов неизвестных, расчета оболочек ядерных реакторов, других высокоточных задач, для их решения необходимо применять современные компьютерные технологии с заданными характеристиками качества по точности и быстродействию [1]. С этой точки зрения, приведенные алгоритмы можно рассматривать как резерв оптимизации вычислений для достижения требуемого качества приближенного решения задачи (шаги 8 – 17 компьютерной технологии решения задач прикладной и вычислительной математики с заданными значениями характеристик качества [1]).

Необходимо также отметить необходимость включения библиотек программ, выполняющих операции над многоразрядными числами, в штатное математическое обеспечение современных многопроцессорных компьютеров, и создание соответствующих программно-аппаратных комплексов [7].

# Выводы

В статье изложены новые алгоритмы и их обоснование для быстрого выполнения операций над многоразрядными числами. Получены оценки сложности.

Приведенные результаты находят применение при решении высокоточных задач, задач информационной безопасности, в математическом обеспечении современных компьютеров.

# Литература


1. Компьютерные технологии решения задач прикладной и вычислительной математики с заданными значениями характеристик качества / В.И. Сергиенко, В.К. Задирака, М.Д. Бабич [и др.] // Кибернетика и системный анализ. – 2006. – № 5. – С. 33-41.
2. Задірака В.К. Комп'ютерна арифметика багаторозрядних чисел / В.К. Задірака, О.С. Олексюк. – Київ, 2003. – 263 с.
3. Николаевская Е.А. Программно-алгоритмические методы повышения точности компьютерных решений / Е.А. Николаевская, Т.В. Чистякова // Кибернетика и системный анализ. – 2009. – № 6. – С. 172-176.
4. Задірака В.К. Аналіз складності швидких алгоритмів обчислення дискретної згортки / В.К. Задірака, С.С. Мельникова // Вісник НУ, серія кібернетики. – 2001. – С. 40-44.
5. Задирака В.К. Теория вычисления преобразования Фурье / Задірака В.К.– Киев : Наук. думка, 1983. – 213 с.
6. Карацуба А.А. Умножение многоразрядных чисел на автоматах / А.А. Карацуба, Ю.П. Офман // ДАН СССР. – 1962. – 145. – С. 293-294.
7. Терещенко А.Н. Умножение больших N-разрядных чисел с использованием только N-разрядных ДПФ / А.Н. Терещенко // Компьютерная математика. – 2008. – № 1. – С. 122-130.
8. Березовский А.И. О тестировании быстродействия алгоритмов и программ выполнения основных операций для асимметричной криптографии / А.И. Березовский, В.К. Задирака, Л.Б. Шевчук // Кибернетика и системный анализ. – 1999. – № 5. – С. 61-68.
9. Задирака В.К. Построение программно-аппаратных комплексов арифметики сверхбольших чисел / В.К. Задирака, А.М. Кудин // Компьютерна математика. Оптимізація обчислень : збірник наукових праць НАНУ. – Київ : Ін-т кібернетики ім. В.М. Глушкова. – 2001. – Т. 1 – С. 158-163.



***В.К. Задірака, А.М. Терещенко***

**Використання арифметики багаторозрядних чисел у сучасних комп'ютерних технологіях розв'язання задач трансобчислювальної складності**

У роботі наводиться ряд оригінальних ефективних за швидкодією алгоритмів виконання операцій над багаторозрядними числами та їх використання для розв'язання задач двоключової криптографії і високоточних обчислень за допомогою комп'ютерної технології розв'язання задач прикладної і обчислювальної математики із заданими значеннями характеристик якості за точністю та швидкодією.



***V.K. Zadiraka, A.N. Tereshchenko***

**Usade of Multidigits Calculation in Modern Computer Technique of Supercalculated Sophisticated Task Solving**

The several processing speed of multidigits calculation algorithms are presented. It is shown the using of those algorithms for solving tasks of two-key cryptography and high-precision calculations using the task solving computer technique of applied and computing mathematics with given quality measure by precision and processing speed.


*Статья поступила в редакцию 29.06.2010.*